Experience Report Summary: "Applying Adaptive Safety Analysis Techniques"

Robyn R. Lutz and Hui-Yin Shaw
Jet Propulsion Laboratory

March 31, 1999

Current needs for high-reliability, reusable software; rapid, evolutionary development; and verification of innovative software architectures have focused attention on improving techniques for analyzing the safety and reliability of embedded software. In particular, the successful application of safety-analysis techniques on such projects depends on our ability to adapt existing methodologies to changing development processes.

Safety analysis techniques for high-reliability systems are widely available. However, these methods tend to emphasize full-scale, thorough, "one-size-fits-all" investigation of a system. When a project must back off from this paradigm, due to cost, schedule, or personnel constraints, guidelines are lacking. The work described here addresses this problem by investigating the adaptive use of software safety techniques for the integrated analysis of software and system safety. The types of adaptation include adaptation to leverage existing documentation; adaptation to current project concerns; adaptation to component-based, embedded software; and dynamically adapting the next analysis step based on the results of prior phases.

The work reported here integrates two successful safety analysis techniques which have been used separately on software and hardware into the system hazard analysis. This process combines Software Failure Modes and Effects Analysis (SFMEA) and Software Fault Tree Analysis (SFTA) in a way that can be readily adapted to a particular project's evolving system needs. The adaptive, integrated approach was used on two recent space instruments: the Mars Microprobe Project and the Earth Orbiting System's Microwave Limb Sounder. Descriptions of related work, of the applications, of sample results, of an experimental web-based support tool, and of the significance and consequences of the results are summarized in the experience report.

Finally, the main lessons learned from this experience are discussed. These attributes of the adaptive verification are recommended elements of any similar process: (1) flexible use, (2) a risk-driven rather than sequential approach, (3) "zoom-in/zoom-out" use, (4) SFMEA and SFTA as complementary techniques, (5) preserving traceability, and (6) applicability to fault protection software.

Applying Adaptive Safety Analysis Techniques

Robyn R. Lutz* and Hui-Yin Shaw †
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109-8099

March 31, 1999

Abstract

Current needs for high-reliability, reusable software; rapid, evolutionary development; and verification of innovative software architectures have focused attention on improving techniques for analysing the safety and reliability of embedded software. The work reported here integrates two successful safety analysis techniques which have been used separately on software and hardware into the system hazard analysis. This process combines Software **Failure Modes and Effects Analysis** (SFMEA) and Software **Fault Tree Analysis** (SFTA) in a way that can be readily adapted to a particular project's evolving system needs. The technique was used on two recent space instruments: the Mars Microprobe Project and the Earth Orbiting System's Microwave Limb Sounder. The main lessons learned from this experience are discussed: (1) flexible use, (2) a risk-driven rather than sequential approach, (3) "zoom-in/zoom-out" use, (4) SFMEA and SFTA as complementary techniques, (5) preserving traceability, and (6) applicability to fault protection software.

Keywords: **software safety, FMEA, FTA, design verification, reliability**

*Contact information for first (contact) author: Dept. of Computer Science, 226 Atanasoff Hall, Iowa State University, Ames, IA 50011-1041; email: rlutz@cs.iastate.edu; phone: 515-294-3654; fax: 515-294-0258.

†Contact information for second author: Jet Propulsion Laboratory, MS 125-233, 4800 Oak Grove Dr., Pasadena, CA 91109-8099; email: hui-yin.shaw@Jpl.nasa.gov; phone: 818-354-9540; fax: 818-393-1362.

1. Introduction

This paper describes the integration of two software safety techniques, Software Failure Modes and Effects Analysis (SFMEA) and Software Fault Tree Analysis (SFTA) into the system engineering process. The application of these techniques on two spacecraft instruments and experience from their use are presented. This work is an extension of previously reported work on integrating software and system safety [9].

The motivation for the work described here derives from the changing needs of the project developers. Current needs for higher reliability, rapid development, reusable software, and innovative software architectures have focused attention on improving design verification techniques. In particular, the successful application of safety-analysis techniques on such projects depends on our ability to adapt existing methodologies to changing development processes.

Safety analysis techniques for high-reliability systems are widely available. However, these methods tend to emphasize full-scale, thorough, "one-size-fits-all" analysis of a system. When a project must back off from this paradigm, due to cost, schedule, or personnel constraints, guidelines are lacking. The work described here addresses this problem by investigating the adaptive use of software safety techniques for the integrated analysis of software and system safety. The types of adaptation, described in Section 3, adapt analyses to existing documentation; to current project concerns; to component-based, embedded software; and dynamically adapt the next analysis step to results of prior analysis phases.

The adaptive use of safety analysis techniques is part of a growing interest in software engineering practices that target high-risk or high-usage areas for attention rather than applying a practice uniformly to all areas or components. The fact that these practices are sometimes called "good-enough" (e.g., "good-enough testing") identifies the challenge to critical systems. On the one hand, high-reliability systems may not be able to afford the risk that "good-enough" practices entail. On the other hand, a careful and prioritized application of these adaptive analytical techniques may yield higher reliable components than other options available to small projects with tight cost and schedule constraints. This paper reports the application of these adaptive practices to components on two space instruments, the New Millenium Program Mars Microprobe Project (MM) and the Earth Observing System's Microwave Limb Sounder (MLS) [2, 13].

The rest of the paper is organized as follows. Section 2 presents related work and explains the design analysis techniques on which this work is based. Section 3 describes the four types of adaptation which were applied to the integrated software and system analyses. Section 4 provides details of the experience and presents the results. Section 5 summarizes the lessons learned from these applications and extracts some key elements which are recommended for any similar process.

2. Related Work

The contribution of this paper is to report experience with the adaptive application of integrated software and system safety techniques. The techniques that this work is based on are well-documented and widely used. Consequently, only brief descriptions with pointers

to further work are given here.

FMEA (Failure Modes and Effects Analysis) has been widely used for design analysis in military and industrial applications since the mid-sixties. Several FMEA standards exist, including a U.S. Military standard, a NASA standard, and a SAE (Society of Automotive Engineers) standard. See [11] for a description and bibliography.

SFMEA (Software Failure Modes and Effects Analysis) is a design analysis method that explores the effects of possible software failure modes on the system. SFMEA is an extension of hardware FMEA. SFMEA has been used on flight projects at Jet Propulsion Laboratory, primarily to verify the correct functioning of system-level fault protection software.

Briefly, SFMEA is a structured, table-based process of discovering and documenting the ways in which a software component can fail and the consequences of these failures. It is most frequently used during the design phase, but has also been used during the requirements phase. The SFMEA process is guided by a set of standardized failure modes (e.g., "Wrong timing of data," "Abnormal process termination") which the analyst considers in turn. The SFMEA is a form of forward (bottom-up) analysis. The process traces the propagation of anomalies from causes (failure modes) to local (subsystem or component) effects to global (system) effects [7, 10, 16].

FTA (Fault Tree Analysis) is a hazard analysis technique that works top-down from an identified undesirable event or hazard to discover its possible causes [5, 7, 19]. FTA uses Boolean logic to decompose an undesirable event into the preconditions that led to the event's occurrence. It is widely used in the systems and hardware areas [1, 6].

SFTA (Software Fault Tree Analysis) uses a similar method to analyze software code or detailed design [8]. Rushby identifies as the goal of SFTA, "to show that a specific software design will not produce system safety failures or, failing that, to determine the environmental conditions that could lead it to cause such a failure" [18].

Some researchers have performed SFMEA as a preparatory activity to fault tree construction [15]. Others have recommmended first performing a search for causes (as in a FTA) and then considering the effects of each failure (as in a FMEA) [17]. Combining forward and backward analyses, or the bottom-up SFMEA with the top-down SFTA, has been found to be effective in understanding underlying combination of circumstances that enable a failure mode to occur, as well as the likelihood of the identified failure mode [4, 10].The effectiveness of the SFMEA is also increased by integrating it with existing system FMEA or system FTA.

3. Adaptive Safety Analysis Techniques

The application of the integrated software and system analyses to the two instruments is described in Section 4. Here we introduce the four types of adaptation that these applications entailed.

The investigations performed in these applications used four types of adaptive analyses:

1. Adaptation to *existing documentation.*
   In both projects, the analyses leveraged off existing documentation. For example, an existing system-level fault coverage matrix was chosen as the initiating point of the safety analysis for three components on Mars Microprobe (MM). Those faults handled by software then served as the failure modes of the high-level SFMEA. On Microwave

Limb Sounder(MLS), where component-level FMEAs and FTAs already existed, the natural choice was to further extend the leaf nodes that involved software into more detailed SFTAs.

2. Adaptation to *specific project concerns.*
   In both projects, the developers were able to point to areas or scenarios which merited additional attention. In general, the developers wanted further assurance that a failure of the fault monitoring and recovery software could not prevent (although it might degrade) mission success. Verification of fault coverage, i.e., that the software assigned to handle known fault scenarios did so reliably, was useful to the MM developers. On MLS, the developers were interested in software interactions that could affect the components in unforeseen ways.

3. Adaptation to *component-based, heavily embedded software.*
   On both projects, the software was heavily embedded in hardware components that involved leading-edge technology. The software was highly coupled with the hardware, monitoring and controlling it. On both projects, correct functioning of the components was critical to the mission's success. In general, attention during early development had focused primarily on defining the hardware and the hardware/software interfaces, rather than on designing the software. As a consequence, we used techniques that could integrate the system and software analyses. The emphasis was on investigating the software that monitors and responds to system failures, the contributing software causes of system failures, and the system effects of software failures.

4. Adaptation to let *prior results drive the next analysis phase.*
   The next step of an analysis of a component was chosen dynamically based on the concerns remaining at the end of the previous stage of the analysis. For example, on MLS the existence of common software failure modes in the FMEA drove the choice of a SFTA as the next step. The SFTA used these common failure modes as root nodes and expanded each sub-node until a basic fault event was reached or no further analysis could be performed.

Figure 1 summarizes the integration of software and system analyses. The Fault Coverage Matrix at the top of the figure was used by MM to document system failures and the software or hardware responsible for their avoidance or recovery. The second row of Fig. 1 shows how FTAs and SFTAs link failure causes and failure events, while FMEAs and SFMEAs link failure modes and failure effects. Both SFTAs and SFMEAs were used to evaluate the software's robustness against failure.

The heavy lines in Fig. 1 show the direction of the analyses performed on one or more of the components. The dashed lines show how the results of the analyses provide verification or insight into fault mitigation and fault handling strategies. In general, adaptive analyses proceeded downwards in the figure from system to component to software emphases. Techniques on the right side of the figure tended to be used for verification of design coverage; techniques on the left side for understanding of design interactions. Horizontal movement (e.g., from SFMEA to SFTA) has been established elsewhere as a useful way to combine the strengths of a forward and backward search, but was not performed here since the emphasis was on integrating the software and the system analyses [12].

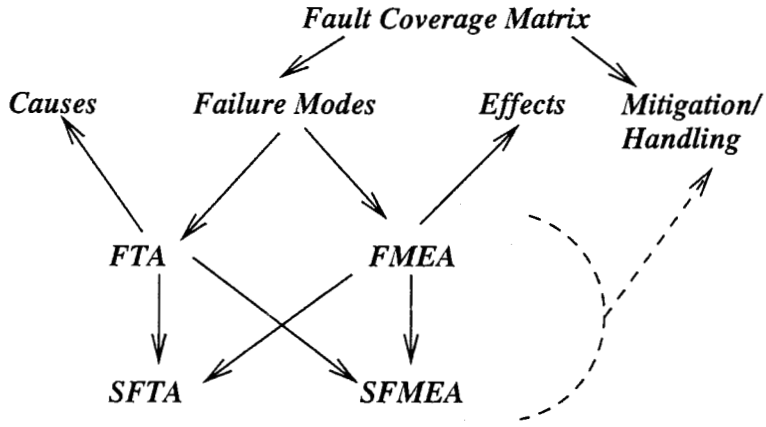

Figure 1: Overview of Analysis Elements

4. Results

This section describes the results of the adaptive applications of the integrated safety analyses to the New Millenium Program Mars Microprobe Project (MM) and the Earth Observing System's Microwave Limb Sounder (MLS). Due to space limitations, only sample results are presented here. See [9] for a fuller description of the results.

4.1 Mars Microprobe

The Mars Microprobe Project (MM), also known as the Deep Space 2 (DS-2) mission, consists of two identical microprobes that will penetrate the Martian surface [13, 14]. MM launched onboard the Mars Polar Lander in January, 1999. The mission will validate technologies that will enable future planetary network missions (e.g. simultaneous deployment of multiple landers, penetrators, etc.) while at the same time collecting meaningful science data on Martian soil conductivity, meteorology, and subsurface ice.

We worked with three MM components that are essential for mission success: Impact Detection (to detect the landing on Mars and penetration of the microprobe for initiation of science activities), the Water Experiment (to detect possible subsurface ice in a soil sample), and Telecommunications (to send data from the microprobe back to the orbiter for transmission to Earth).

On MM, the project had already produced a system-level fault-coverage table, which we used as the failure analysis baseline. The table was included in the Mars Microprobe Spacecraft Design document [14]. The table, entitled "Fault Coverage," listed for each key function (e.g., telecommunications) the types of faults that could occur (e.g., loss of uplink, loss of downlink, etc.) and the coverage that is provided for each of these fault types. Some of the fault types involved hardware failure; some involved software failure; and some involved both. The coverage for some faults was, at least in part, software-based. Response to these faults usually entailed software control of hardware devices. Onboard fault protection software exists to aid in recovery from failures during the science phase of the mission.

4.1.1 Impact SFMEA

The initial analysis sought to verify the adequacy of the software handling of the fault types described in the system-level fault coverage table. We did this by performing SFMEA for the three critical functions described above. The objectives included identifying critical software failures and assessing the appropriateness of the fault avoidance and mitigation and of the recovery sequence. For example, one of the MM components studied was the Impact Detection. The impact-related fault monitor and response as well as the event sequence (consisting of the Impact Loss Monitor, the Accelerometer-Off Response, and the Impact Detection and Penetration Measurement) were evaluated.

This study yielded the following results: (1) clarification of sequence and fault response and clarification of a software variable in an anomalous event; (2) recommendation of test cases; and (3) identification of a software fault avoidance strategy. Some key questions involving unclear definitions and missing information for anomalous scenarios also surfaced during this study. These findings were considered by the Design Engineer for inclusion in a future document update. Several failure entries marked in the SFMEA as medium to high criticality failures were identified as verifiable in test (i.e., tests can be performed to verify the absence of potential faults). Recommendations were then made to the Project to include these test cases.

Most of the highly critical failures involved software hang-up (halt). Depending on when in the event sequence the hang-up occurs, such hang-ups can cause missed science experiment data (even with fault protection and recovery sequences in working order). A discussion with the software developer (a domain expert) was conducted to identify sources leading to possible software hang-up. It turned out that a potential software hang-up exists when a particular assembly command is executed at the same time that a certain built-in interrupt timer goes off. Therefore, the software hang-up may be sporadic and difficult to debug. A fault avoidance method was recommended by the domain expert to alleviate this problem.

4.1.2 Integrating System and Software FMEAs

A subsystem (component-level) FMEA table was constructed for the Telecom system with the information obtained from the system design document [14]. This exercise showed that performing FMEA on a critical (sub)system, component, or function could help identify areas requiring fault monitor and response modules. These monitor and response modules are identified in the Failure Detection/Correction column of the FMEA table.

The subsystem FMEA table and the subsequent fault monitor and response design lay the groundwork for the top-level (requirements/design-level) software safety analyses. In our study, three essentail software-controlled monitor and response modules from the Telecom subsystem FMEA were expanded into the top-level SFMEA.

At the time when we were conducting this SFMEA study, the MM Project was transitioning into test phase. As it turned out, all the highly critical software failures identified in this study were verifiable in test for software implementation correctness. Therefore, recommendation was made to the project for inclusion of these failure scenarios in their test cases.

Some of the lessons learned during the application of the integrated safety techniques

on MM were considered to be transferable technology. These lessons are summarized in Section 5 and formed part of the input to the process for MLS analysis. Since MLS is at an earlier phase of development than MM, the system-level analysis performed to date has been primarily hardware-oriented. Techniques such as SFMEA that focus on the software's contribution to system fault coverage can be used to expand analysis of critical components or capabilities.

4.2 Microwave Limb Sounder

The Earth Observing System Microwave Limb Sounder (MLS) instrument, currently under development, will support an investigation to improve understanding and assessment of stratospheric ozone depletion and chemistry, tropospheric ozone distribution and chemistry, and climate change and variability [2, 3]. The MLS instrument will measure naturally occurring microwave thermal emission from the limb of Earth's atmosphere to remotely sense vertical profiles of selected atmospheric gases, geopotential height, temperature and pressure. The MLS instrument will fly on the EOS Chemistry platform to be launched in December, 2002.

In the MLS project, the component-level FMEAs were reviewed for potential failures where software might play a part. It was found that similar failure types appeared among the component-level FMEAs. For example, the "Loss of Bus Synchronization" failure mode appears in two component FMEAs. When appropriate, we generalized these common failures when performing the top-level SFTA. Each of the selected failures became the top-level hazardous event (root node) of a SFTA. For each root node hazard, we worked backwards (top-down), expanding each sub-node until a basic event was reached (a leaf of the fault tree), or until no further analysis could be done.

A discussion with the software engineer on the SFTAs yielded several follow-up items and a few proposals for further analyses. Feedback from the software engineer was incorporated into the final SFTA. The software engineer felt that this was a worthwhile exercise in evaluating all the possible failures/faults and their avoidance and mitigation.

Four MLS component-level FTAs were also reviewed. Faults (or leaf nodes of Fault Trees) that may be attributed to software failure were identified. These selected component faults became the root node hazard (or the root of a fault tree) for the next lower-level FTA, in this case, the top-level SFTA. The same SFTA procedure was performed as in the component-level FMEA to top-level SFTA study.

In system and component level FMEAs and FTAs, the analyses are most frequently hardware-parts oriented. Software and operational attributed faults at the system or component level are often left out. The MLS top-level SFTAs also validated the adequacy of software commands for the control of hardware mechanisms.

The SFTAs identified the following types of fault tree leaf nodes:

- Software faults verifiable in test: e.g., command format error, telemetry transmission scheme compatibility
- Lower-level (source code) analysis required: e.g., the need to determine software/mechanism behavior resulting from out-of-range command parameters

- Operational errors: e.g., incorrect command sequence
- Hardware-attributed faults: e.g., electronic noise-induced command bit drop in the bus
- External faults: e.g., spacecraft telemetry pickup error.

Additional findings from the top-level software FTA include: (1) Further analysis required: e.g., instrument/software behavior resulting from executing a command in an inappropriate mode; (2) Follow-up cases identified: e.g., to determine an appropriate reset mechanism; (3) Workarounds identified: e.g., capability for in-flight software update to respond to faulty register memory map.

4.3 Web-Based Support

Data from safety analysis can become massive and unmanageable. To manually sift through these data for specific information is tedious and carries the risk of inadvertently overlooking critical data.

A web-based database application can link upper-level safety analyses to lower-level analyses. For example, a system FMEA can be linked to a component-level FMEA, and these can be linked to specific software and hardware FMEAs. This allows better traceability of safety-critical elements from the system to the software and the hardware, and to their fault avoidance or mitigation strategies. Depending on the project's needs, links to safety requirements, tests, event sequences or design can also be implemented. The tool can also provide options to perform searches in specific areas of interest, such as failure criticality, failure modes, or affected requirements. Restricted editing capabilities can support on-line updates.

In an experimental tool development, Paul Davis created dynamic Web pages that interacted with the FMEA database. This web-based safety analysis application stores the FMEA results and provides the ability to do search and report on the FMEA entries. For example, a subsystem FMEA for Telecom is stored as one table, and a software FMEA is stored as another table. A user can search on criticality, testability, and/or failure modes in any selected-combination of tables. In retrospect, an additional search option that would be useful is "Affected Requirements."

The experimental web-based tool demonstrated that we could quickly locate the information we needed to help us do our work. For example, we could identify failures that were identified as testable in the earlier safety analyses and use this information to follow up in test planning and test verification when appropriate. Most importantly, the safety analysis results can be readily accessible to project developers and analysts for follow-up and further analysis of critical issues. Similarly, safety-critical information can be easily available for anomaly impact analyses and for requirements or design change impact assessments.

5. Conclusions

The lessons learned in the adaptive application of the integrated software and system safety techniques are summarized below. These are recommended elements of any similar process.

1. *Flexible use.*
   We found that a key advantage of the integrated approach is that the focus of the analysis can be tailored to the needs, phase, and available documentation of the specific project. On MM this meant using the existing hazards analysis work that had been done as a baseline and extending the analysis (via SFMEA) in the directions that were of most concern to the project.

2. *Risk-driven*
   An advantage of the integrated SFMEA/SFTA approach is that it allows a dynamic re-focusing of attention to allow prior analysis results to drive the next analysis phase. For one component, a SFMEA followed by a verbal walkthrough with experts resolved the open issues. For another component, a SFMEA was later supplemented by a FTA to follow up on some issues of concern. SFMEA and/or SFTA can be performed only for those components perceived as possibly presenting unacceptable risk, or SFMEA/SFTA can be applied selectively to differing levels of detail on different components, all depending on the project's needs.

3. *"Zoom-in/zoom-out" use of SFMEA/SFTA*
   A consequence of the flexible use of SFMEA/SFTA is that it can provide a "zoom-in/zoom-out" approach to analysis of critical components. Selective targeting of issues of concern, designs that have changed, or areas that raise unresolved questions is possible. A "zoom-in" can be chosen to examine more closely a particular piece of the system or the effect of a particular scenario. Similarly, a "zoom-out" can be chosen to examine a wider piece of the system or the interfaces. On MM, for example, tuning the analysis to the evolution of the system meant that questions arising during the initial analysis of one component (Impact Detection) led both to a quick "zoom-out" review of the system-level interface FMEA (to check if there were any related software issues) and to a "zoom-in" on the SFMEA issue regarding software hang-up (to investigate the fault avoidance strategy).

4. *SFMEA and SFTA as complementary techniques*

   SFMEA and SFTA have a well-deserved reputation as complementary techniques (see Section 2). In particular, the combination has been used to identify unexpected dependencies and interactions in the system. On MM we primarily performed SFMEA, since the project had already identified the critical faults that needed coverage in the system. and the causes of the faults were nearly all hardware or environmental (e.g., landing) failures. On MLS, we supplemented the FMEA with the SFTA in order to trace the possible software involvement in failures.

5. *Preserving traceability*

   SFMEA/SFTA can be performed on a system at varying degrees of detail, so traceability between higher-level and lower-level analyses must be maintained. Sometimes the traceability among levels is explicit. For example, failure effects in a lower level FMEA may be the failure modes in the left hand column of a higher-level FMEA. Similarly, for FTA, a single node in a high-level tree may be decomposed into a more detailed

FTA. Web-based support, such as that described in Section 4.3, enhances our ability to manage the traceability.

6. *Applicability to fault protection software*

   In this study, our interest was in SFMEA/SFTA as safety analysis techniques, so we chose critical fault protection software for the applications. SFMEA is especially well-suited to analysis of fault protection monitors and responses. For monitoring software, SFMEA was used to check that false-positives were not produced and that adequate reasonableness checks were performed on the values used to make control decisions. For fault protection software that responds to faults, SFMEA was used to check that the effect of the response (e.g., reconfiguration, try-again, etc.) matched the intent of the fault response. By dynamically adapting the integrated software and system safety techniques to the needs and realities of the two projects, the analyses provided some assurance that the fault coverage specified in the design document was adequate and robust.

Acknowledgments

We thank Sarah Gavit, Kari Lewis, Parviz Danesh, Bob Detweiler, and Robert Nowicki on MM; Gary Lau, Dennis Flower, Marc Walch, Mike Girard, Mark Boyles, Philip Szeto, John Klohoker, and Johnathan Carson on MLS; and Paul H. Davis for assistance with the web page application.

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Funding was provided under NASA's Code Q Software Program Center Initiative, UPN #323-08.

Reference herein to any specific commercial product, process, or service by tradename, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

References

[1] DeLemos, R., A. Saeed, and T. Anderson, "Analyzing Safety Requirements for Process-Control Systems," *IEEE Software*, Vol. 12, No. 3, May, 1995, pp. 42-52.

[2] "Earth Observing System, Microwave Limb Sounder Project Homepage," *http : //mls.jpl.nasa.gov/*

[3] Earth Observing System Microwave Limb Sounder, "System-Level Failure Modes, Effects, and Criticality Analysis," JPL D-16088, Version 1.0, September 1, 1998.

[4] Fenelon, P. and J. A. McDermid, "An Integrated Toolset for Software Safety Assessment," *Journal of Systems and Software*, July, 1993.

[5] Hansen, K.M., A.P. Ravn, and V. Stavridou, "From Safety Analysis to Software Requirements," *IEEE Transactions on Software Engineering*, Vol. 21, July, 1993, pp. 279-290.

[6] Knight, J. and L. G. Nakano, "Software Test Techniques for System Fault-Tree Analysis," 1997.

[7] Leveson, N., *Safeware, System Safety and Computers*, Addison-Wesley, 1995.

[8] Leveson, N., S. S. Cha, and T. J. Shimeall (1991), "Safety Verification of Ada Programs Using Software Fault Trees," *IEEE Software*, July, pp. 48–59.

[9] Lutz, R. and H.-Y. Shaw, "Applying Integrated Safety Analysis Techniques (Software FMEA and FTA)'," JPL D-16168, Nov. 30, 1998.

[10] Lutz, R. and R. Woodhouse (1997), "Requirements Analysis Using Forward and Backward Search," Annals of Software Engineering , 3, 459–475.

[11] Lutz, R. and R. Woodhouse, " Failure Modes and Effects Analysis," *Encyclopedia of Electrical and Electronics Engineering*, ed. J. Webster, John Wiley and Sons Publishers, 1998.

[12] Lutz, R. and R. Woodhouse, "Bi-Directional Analysis for Certification of Safety-Critical Software," *Proceedings of the First International Conference on Software Assurance Certification*, Chantilly, VA, March, 1999.

[13] Mars Microprobe Project homepage, *http : //nmp.jpl.nasa.gov/ds2/*

[14] Mars Microprobe, *Spacecraft Design*, JPL D-14222, Rev. A, Nov. 13, 1997 and Rev. B Draft, Jan. 22, 1998.

[15] Maier, T., "FMEA and FTA To Support Safe Design of Embedded Software in Safety-Critical Systems," in *CSR 12th Annual Workshop on Safety and Reliability of Software Based Systems*, Bruges, Belgium, 1995.

[16] Mazur, M. F., "Software FMECA," *Proc. Fifth International Symposium on Software Reliability Engineering*, 1994.

[17] McDermid, J. A., M. Nicholson, D. J. Pumfrey, and P. Fenelon, "Experience with the application of HAZOP to computer-based systems," in *Proc of COMPASS '95*, Gaithersburg, MD, 1995, pp. 37-48.

[18] Rushby, J., *Formal Methods and Digital Systems Validation for Airborne Systems*, SRI-CSL-93-07, 1993.

[19] Storey, Neil. *Safety-Critical Computer Systems*. Addison-Wesley, 1996.